AMENDMENTS TO THE CLAIMS

This listing of claims replaces all prior versions, and listings, of claims in the application.

1. (Previously presented) A method for partitioning a specification in a source code, characterized in that the method comprises the following steps:

first converting the specification into a plurality of abstract syntax trees;

partitioning the plurality of abstract syntax trees into at least a first set and a second set of abstract syntax trees, the first set of abstract syntax trees to be implemented by a first processor and the second set of abstract syntax trees to be implemented by a second processor; and

second converting, after the partitioning step, the first set of abstract syntax trees and the second set of abstract syntax trees back to source code, thereby facilitating editing of a resulting specification in the source code.

2. (Original) A method for partitioning a specification in a source code according to Claim 1, wherein the second processor is a co-processor.

3. (Original) A method for partitioning a specification in a source code according to Claim 2, wherein the first processor is a general-purpose processor.

4. (Previously presented) A method for partitioning a specification in a source code according to Claim 1 wherein the second converting step comprises converting the first set of abstract syntax trees to a first partial specification in the source code and converting the second set of abstract syntax trees to a second partial specification in the source code.

5. (Currently amended) A method for partitioning a specification in a source code according to Claim 1 wherein the step of partitioning the plurality of abstract syntax trees into a first set of abstract syntax trees and a second set of abstract syntax trees comprises a step of out-lining at least one abstract syntax tree of the first set of abstract syntax trees based on profile data and replacing said abstract syntax tree by a function call to a function present as an abstract syntax tree in the second set of abstract syntax trees.

6. (Currently amended) A method for partitioning a specification in a source code according to Claim 1 wherein the step of partitioning the plurality of abstract syntax trees into a first set of abstract syntax trees and a second set of abstract syntax trees comprises a step of out-lining at least one abstract syntax tree of the first set of abstract syntax trees based on programmer provided information and replacing said abstract syntax tree by a function call to a function present as an abstract syntax tree in the second set of abstract syntax trees.

7. (Previously Presented) A co-design method for producing a target system wherein the target system comprises a first processor and at least a second processor:

the co-design method comprising the method for partitioning a specification in a source code according to Claim 1.

8. (Previously presented) A co-design method for producing a target system according to Claim 7, wherein the second converting step of the method for partitioning a specification in a source code comprises converting the first set of abstract syntax trees to a first partial specification in the source code and converting the second set of abstract syntax trees to a second partial specification in the source code.

9. (Previously presented) A co-design method for producing a target system according to Claim 8 wherein the second processor is a co-processor and wherein the second partial specification is converted to a specification of the co-processor.

10. (Currently amended) A co-design method for producing a target system according to Claim 9, wherein the first processor is a general-purpose processor and wherein the first partial specification is converted to object code by means of a compiler.

11. (Previously presented) A co-design method for producing a target system according to Claim 10, further comprising a step for defining an interface between the general-purpose processor and the co-processor.

12. (Original) A co-design method according to Claim 9, wherein the specification of the co-processor comprises a specification of an ASIC.

13. (Original) A co-design method according to Claim 9, wherein the specification of the co-processor comprises a specification of a programmable processor.

14. (Original) A co-design method according to Claim 9, wherein the specification of the co-processor comprises a specification of a reconfigurable processor.

15. (Previously presented) A co-design method according to Claim 11, wherein the interface between the general-purpose processor and the co-processor comprises a remote function call;

the remote function call having a set of parameters;

the set of parameters comprising an identifier for the function to be called, at least one reference pointing to the input data of the function to be called and at least one reference pointing to the result data of the function to be called.

16. (Original) A co-design method according to Claim 15 wherein the set of parameters of the remote function call further comprises a reference to a memory location used for storing information on the return status of the function to be called.

17. (Previously presented) A co-design method according to Claim 7 wherein the target system further comprises a system memory and a system bus ;

the system memory, the first processor and the second processor being coupled by the system bus.

18. (Previously presented) A co-design method according to Claim 10 wherein the general-purpose processor is a digital signal processor.

19. (Canceled)